Universidad de San Carlos de Guatemala
Escuela de Ingeniería en Ciencias y Sistemas, Facultad de Ingeniería
Introducción a la programación y computación 2, 1er. Semestre 2022.

# PROYECTO 3

**202000173 – Christian Alessander Blanco González**

**Resumen**

El programa consiste en poder crear una aplicación web que esta conformada por Backend y Frontend. Para poder obtener la conexión de ambos fue se necesitó de una librería llamada Django, la cual ya trae incluido un método llamado jinja que la cual jinja nos ayudó para la creación de los forms de cada pagina para nuestro proyecto y Django fue la encargada de hacer la conexión HTTP de nuestro Frontend hacia nuestro Backend por medio de request.

**Palabras clave**

**Frameworks:** Un entorno de trabajo o marco de trabajo es un conjunto estandarizado de conceptos, prácticas y criterios para enfocar un tipo de problemática particular.

**API: e**s un conjunto de definiciones y protocolos que se usa para diseñar e integrar el software de las aplicaciones.

**Frontend:** es la parte del desarrollo web que se dedica a la parte frontal de un sitio web, en pocas palabras del diseño de un sitio web.

**Backend:** es la parte del desarrollo web que se encarga de que toda la lógica de una página web funcione.

**HTTP:** es el protocolo de comunicación que permite las transferencias de información a través de archivos en la World Wide Web.

***Abstract***

*The program consists of creating a web application that consists of Backend and Frontend. In order to obtain the connection of both was needed a library called Django, which already brings included a method called jinja which jinja helped us for the creation of the forms of each page for our project and Django was responsible for making the HTTP connection from our Frontend to our Backend via request.*

***Keywords***

***Frameworks:*** *A framework is a standardized set of concepts, practices and criteria for approaching a particular type of problem.*

***API:*** *is a set of definitions and protocols used to design and integrate application software.*

***Frontend:*** *is the part of web development that deals with the front end of a web site, in short the design of a web site.*

***Backend:****. is the part of web development that makes sure that all the logic of a web page works.*

***HTTP:*** *is the communication protocol that allows information transfers through files on the World Wide Web.*

## Introducción

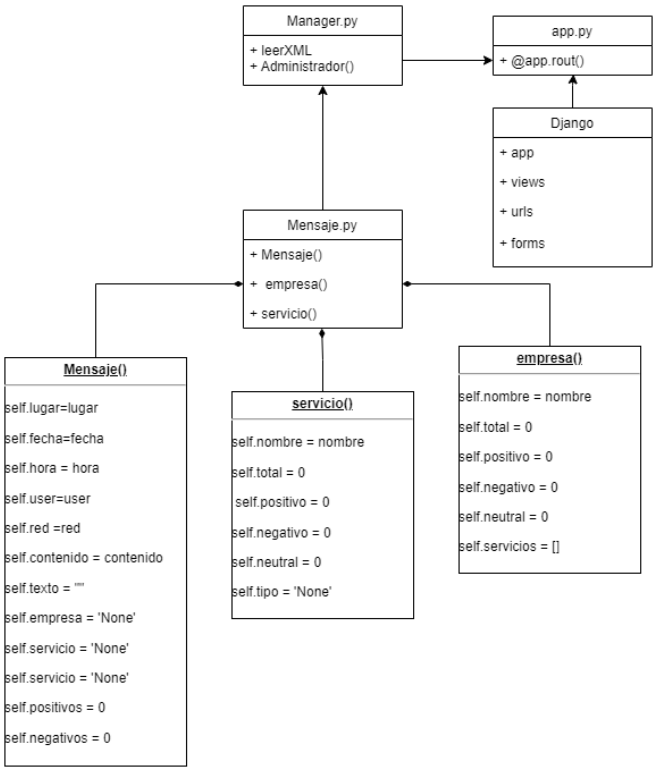
La empresa Tecnologías Chapinas, S.A. está desarrollando una herramienta que sea capaz de analizar contenido de redes sociales y establecer el sentimiento de los usuarios respecto a una empresa y los servicios que provee. Para lograr este fin, la empresa Tecnologías Chapinas, S.A. lee mensajes en redes sociales y construye estructuras de texto con el siguiente un formato establecido.

## Desarrollo del tema

El problema que plantea la empresa "Tecnologías Chapinas, S.A" es el siguiente:

"La empresa Tecnologías Chapinas, S.A. ha creado una estrategia para establecer si un mensaje tiene un sentimiento positivo, negativo o neutro a través de la creación de un diccionario de datos que determine palabras que puedan calificar un mensaje como positivo o negativo, en caso de no tener palabras del diccionario de datos específico, o bien, que la cantidad de palabras con sentimientos positivos y negativos sean iguales, entonces, se considera que el mensaje es neutro. En este mismo diccionario de datos, es posible determinar los nombres de las empresas y sus servicios que se están analizando para determinar si en un momento dado, las redes sociales están mostrando un sentimiento positivo o negativo del mismo."

Para plantear una solución viable a este problema, lo primero que se realizó fue un diagrama de clases, donde nos describe a detalle y gráficamente las estructuras que se implementarán en nuestra aplicación utilizando el famoso paradigma de programación Orientada a Objetos (POO). El diagrama es el siguiente:




Esta aplicación fue desarrollada con uno de los lenguajes de programación multiparadigma conocido como Python, y realizado en el entorno de trabajo de Visual Studio Code. Flask ofrece sugerencias, pero no impone ninguna dependencia o diseño del proyecto. Depende del desarrollador elegir las herramientas y bibliotecas que desea utilizar. Hay muchas extensiones proporcionadas por la comunidad que facilitan la adición de nuevas funciones. Django se basa en el patron de diseño conocido como modelo-vista-controlador (MVC).

Luego de analizar el diagrama de clases, se empezaron a programar las clases necesarias con cada uno de sus atributos y funciones asignados. Para la interacción entre el programa y el usuario se creo una base de html, para que siempre tuviera la misma vista a la hora de realizar diferente acción dentro de la aplicación en la nube.

Para poder leer el archivo XM que contiene las ciudades almacenadas que debemos de analizar, se utilizó una librería llamada xmltodict. Mientras se van leyendo los archivos se van guardando cada uno de los datos extraídos en variables, para luego almacenarlas en listas y ya después guardarlas en el objeto perteneciente.

Luego de que ya se han extraído cada uno de los datos que necesitaremos, creamos nuestro objeto Mensajes() que nos permitirá almacenar cada mensaje con sus respectivos elementos a utilizar para poder realizar la gráfica. Después de haber almacenado cada dato en nuestro objeto procedemos a crear nuestra clase Administrador() que es la encargada de hacer todas las búsquedas y ordenamientos que se solicitan

Para poder obtener los tipos de mensajes por empresas se hizo la creación de la función clasificacionMensaje, esa función dentro de la clase Administrador() es la encargada de poder hacer los conteos de la cantidad de mensajes por su tipo por su empresa y servicio. Luego se hizo la creación de una función llamada salidaXML() que es la encargada de poder mostrarnos en formato XML la salida del archivo de entrada.

Para la realización de la clasificacion de los mensajes por su fecha y empresa se creó la función clasificacionFecha donde recibe dos parámetros que es la fecha y el nombre de la empresa. Esta función funciona de la siguiente manera: primeramente, el parámetro fecha se parseó como una cadena y se le quitaron los guiones, luego de eso se estableció el formato de fecha que se quizo obtener, se creo una lista llamada clasificacion que es una lista temporal ya que en esa lista se estará guardando lo que es son diccionarios con los nombres de la empresa, fecha, positivos, negativos y neutros. Luego se utilizó una serie de if y for anadidos en donde dice que para x variable en nuestra lista de mensajes hacemos una condicional donde decimos que si la fecha de nuestra lista mensajes es igual a la fecha que nos proporciona y que si la empresa de nuestra lista de mensajes es igual a la empresa que se nos proporcionó, procedemos con la realización de recorrer con un for nuestra lista de clasificación donde hacemos cada una de las validaciones para saber el tipo de nuestro mensaje. Luego se hizo la creación de un diccionario para poder guardar los datos obtenidos, y así fuera un poco más fácil el manejo en postman.

Luego tenemos la función clasificacionRango_Fecha, que recibe 3 parametros que es nuestra fecha inicial, fecha final y empresa. Hacemos el mismo procedimiento que se hizo en nuestra función anterior del parseo de la fecha, ya luego con un for accedemos a nuestra lista de mensajes que es un objeto como tal, para así poder acceder a nuestra data, y hacemos las validaciones de que si nuestra fecha inicial

es mayor o igual a nuestra fecha de nuestra lista de mensajes y que si nuestra fecha final es menor igual a nuestra fecha de nuestra lista de mensajes, para luego decir que en todas las empresas y creamos una lista para guardar también nuestros diccionarios al que le pasamos los valores de la variables después. Luego hacemos la validación del tipo de mensaje para cada empresa y servicio, con la excepción de que si el servicio pertenece a esa empresa pues no se sumará. Nuestro diccionario lo metemos a nuestra lista que se debe de crear desde donde inicia nuestra función.

Para poder hacer las peticiones de postman hacia nuestra API, se necesitó de la librería llamada Flask y CORS, la cual estas nos ayudan a poder hacer las peticiones HTTP del exterior hacia nuestra API. La cual se crearon varios endpoint, donde dependiendo el endpoint va a depender el método de envió si será un método POST o método GET. Para poder leer el archivo XML se creo el endpoint /CargaMasiva con el método POST y para obtener lo que es la salida del archivo XML que se cargo se hizo la creación de otro endpoint /consultar que este nos devuelve como tal la estructura de un archivo XML. De igual manera se hizo para la clasificación de mensajes por fecha y empresa donde se creo el endpoint /clasificar-por-fecha con los métodos GET y POST, si el request es get solo imprimios el nombre de la empresa como tal, pero si el request es un POST debemos de ingresar en formato json lo que es la fecha con el nombre fecha y la empresa con el nombre empresa. Para la clasificación de mensajes por intervalo de fecha y empresa se creo el endpoint /resumen-por-rango, donde esta compuesta por los métodos GET y POST, cuando el request es un GET solo se obtiene un listado del nombre de las empresas, y cuando se hace el POST tendremos que ingresar en formato json 3 tipos de variables, donde sería fecha1 que es nuestra fecha inicial, fecha2 que nuestra fecha final y empresa que es el nombre de la empresa. Y por ultimo el endpoint /ayuda nos devuelve lo que son lo datos del creador y la documentación.

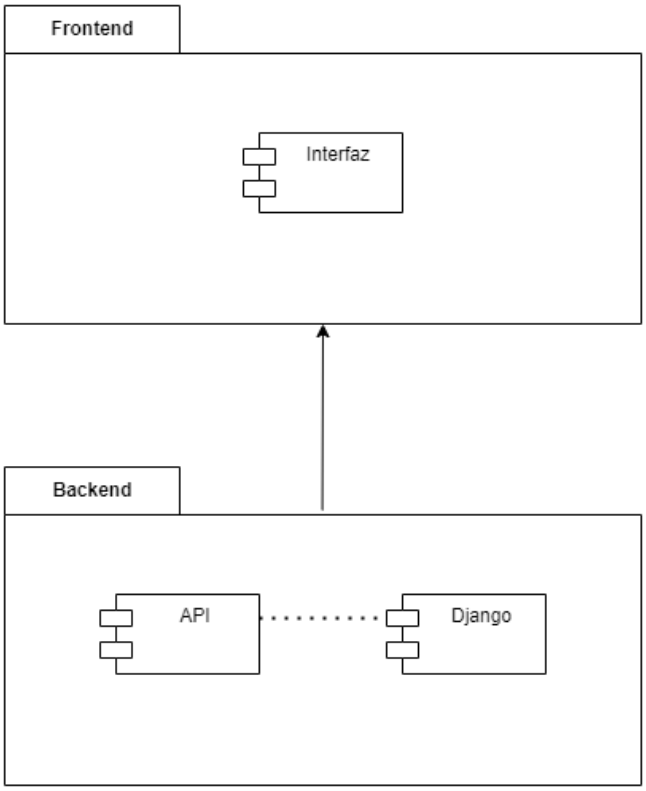
## Conclusiones

El uso de paradigma de programación orientada a objetos facilita la programación de soluciones de esta índole.

La solución planteada se acepta ya que genera un resumen de x cantidad de mensajes que pertenecen a un servicio donde están ligados a una empresa y el poder saber la cantidad de mensajes según su estado.

**Anexos**

Diagrama de componentes, para poder obtener una mejor idea de la implementación o la estructura de nuestro diagrama de clases con nuestro frontend.




**Referencias bibliográficas**


- Docs.djangoproject.com. 2022. Documentación de Django | Documentación de Django | Django. [online] Available at: https://docs.djangoproject.com/es/4.0/ [Accessed 6 May 2022].

- Flask.palletsprojects.com. 2022. Changes — Flask Documentation (2.1.x). [online] Available at: https://flask.palletsprojects.com/en/2.1.x/changes/#version-2-1-2 [Accessed 6 May 2022].

- Docs.python.org. 2022. 3.10.4 Documentation. [online] Available at: https://docs.python.org/3/ [Accessed 6 May 2022].

C. J. Date, (1991). *An introduction to Database Systems.* Addison-Wesley Publishing Company, Inc.